

# From NAND to Tetris in 12 Easy Steps

Shimon Schocken<sup>1</sup> and Noam Nisan<sup>2</sup>

**Abstract** - As CS and EE courses become increasingly more specialized, students are increasingly unable to grasp major ideas that cut across traditional course lines. This workshop presents a course that restores the big picture and demystifies the integrated design and function of computer systems. Using a modular series of 12 projects, students are guided through the gradual construction of a complete working computer system. Starting with simple NAND gates, the students build a general-purpose hardware platform and a modern software hierarchy, yielding a simple but surprisingly powerful computer. This is achieved in a one-semester course by virtue of extreme focus and modular design. The course is completely self-contained, requiring no special equipment or software beyond what is given in the course web site, and is accompanied by a new MIT Press textbook.

*Index Terms* – Computer systems, Computer architecture, Capstone courses, Hardware and software projects.

## WORKSHOP DESCRIPTION

Most computer science students learn different hardware and software themes in different courses, taken at different stages in the program. As a result, they don't gain a gestalt understanding of how computer systems work as integrated enterprises. During the last 5 years we developed a pedagogical approach and a set of software tools that address this challenge by engaging the students in the construction of a complete computer system, from the ground up.

Beginning with nothing more than elementary logic gates, the students gradually build and unit-test a chip-set, a CPU, and a typical computer architecture. Next, we walk them through the development of a modern software hierarchy consisting of an assembler, a virtual machine, a mini operating system, and a compiler for a simple object-based language. The hardware projects are done in a basic Hardware Description Language that can be self-learned in a few hours, and a hardware simulator supplied by us. The software projects can be done in any programming language (in the six times that we taught this course, we received projects in Java, VB, C, C#, and Perl). In order to minimize design uncertainty and facilitate unit-testing of all hardware and software modules, we provide extensive APIs and some 200 test programs, test scripts, and test files. This also helps illustrate working in an exemplary programming-at-the-large setting.

The machine that emerges from this effort is a general-purpose computer that is simulated on the student's

PC and can run games like Tetris, Snake, and other cool programs developed by the students.

An important objective of our approach is to present key hardware and software abstractions learned in other courses and make them concrete through guided implementations. We are able to squeeze all this into a one-semester course framework since we deal with neither efficiency nor advanced features, leaving these important subjects to other courses in the program. The resulting approach is completely self-contained, requiring only programming as a pre-requisite. Hence, courses based on the approach can be given at almost any stage in undergraduate and graduate programs.

Our course web site contains all the materials necessary to run the course, and students and instructors are welcome to use and extend them. The course materials are highly modular and self-contained, meaning the each project can be developed independently. Since the projects can be implemented in any programming language, they can also be viewed as a library of exercises that can be used in other courses as well.

The purpose of this workshop is to present the course and the approach. We will discuss the course's main themes and modules, walk through some illustrative projects, and demonstrate our software tools, which are available in open source and can run on either Windows or Linux. The course is accompanied by a new textbook forthcoming in October 2004. Workshop participants will receive a complementary copy of this book, and a CD with all the course software and materials. For more information see [www.idc.ac.il/tecs](http://www.idc.ac.il/tecs).

<sup>1</sup> Shimon Schocken, Efi Arazi School of Computer Science, IDC Herzliya, Israel, [shimon@idc.ac.il](mailto:shimon@idc.ac.il)

<sup>2</sup> Noam Nisan, School of Computer Science and Engineering, Hebrew University of Jerusalem, Israel, [noam@cs.huji.ac.il](mailto:noam@cs.huji.ac.il)